3.9.1 Método *branch-and-bound* 239

Algoritmo *Branch and Bound*-B&B[1]

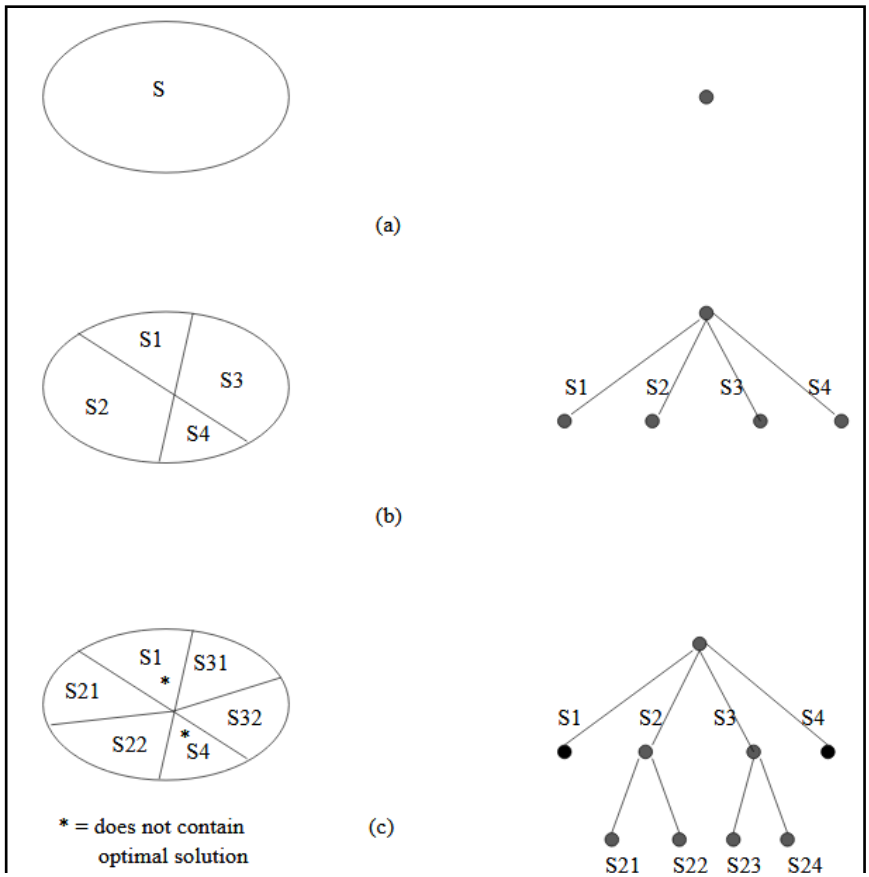
A. H. Land and A. G. Doig. An Automatic method of solving discrete programming problems. *Econometrica,* 28:497 520, 1960)

Os primeiros algoritmos *branch and bound* surgiram na década de 50 como métodos enumerativos na busca de soluções para os problemas de otimização combinatoriais. Naquela época, os pesquisadores perceberam algo, além da eficiência apresentada nos resultados dos experimentos, que o tornaria o algoritmo de otimização mais utilizado: a capacidade de serem descritos independentemente do problema, ou seja, sua generalidade - http://www.din.uem.br/sbpo/sbpo2003/pdf/arq0181.pdf.

Por se tratar de um método enumerativo, a solução é obtida através do particionamento do espaço de soluções, ou seja, o problema original é repetidamente decomposto em subproblemas menores até que uma solução seja obtida. O método aplicado permite identificar tal solução como a melhor possível ou ótima dentre todas as soluções viáveis para o problema.

Para tornar isso possível são construídas, dinamicamente, estruturas de árvore de forma a representar todos os possíveis subproblemas derivados a partir do problema original. Assim, o problema original é localizado na raiz da árvore e dá origem a todos os demais por decomposição.

[1] *Two basic stages of a general Branch and Bound method: i) Branching: splitting the problem into subproblems ii) Bounding: calculating lower and/or upper bounds for the objective function value of the subproblem*



https://imada.sdu.dk/~jbj/heuristikker/TSPtext.pdf

O conjunto de vetores que satisfazem as restrições impostas no modelo define o espaço de soluções. Esse espaço de soluções é organizado através da técnica B&B como um grafo, que usualmente é representado por uma estrutura de árvore. O nó raiz da árvore, também chamado de subproblema inicial, representa o problema original.

O termo subproblema é utilizado para denotar um problema derivado do original através da adição de novas restrições e, dessa forma, corresponde a um subespaço do espaço de soluções original. Portanto, a geração de subproblemas dá-se usualmente pela atribuição de valores a variáveis, o que revela a característica enumerativa do método.

O caminho do nó raiz a um determinado nó folha (nó que não possui filhos) define um elemento do espaço de soluções. Os nós com essa propriedade são chamados de nós solução. Nós solução que satisfazem às restrições impostas no modelo são denominados de nós solução viável ou nós resposta.

Em cada nó da árvore são aplicadas operações específicas do algoritmo com o propósito de verificar se a busca deve avançar através daquele subproblema, o que significa nova decomposição, ou se o nó deve ser cortado. Tais operações ou regras formam a base do algoritmo e revelam sua característica de generalidade, que o torna muito atraente dentre os métodos utilizados.

Um algoritmo B&B para um problema de minimização/minimização consiste, portanto, em três componentes principais:

1. *bounding function*: calcula os limites inferiores e/ou superiores para o valor da função objetivo do subproblema;
2. *strategy for selecting*: uma estratégia para selecionar o subespaço de soluções a ser investigado na iteração atual; e
3. a *branching rule*: regra de ramificação a ser aplicada se um subespaço, após a investigação, não puder ser descartado, subdividindo o subespaço em dois ou mais subespaços a serem investigados em iterações subsequentes.

O nó raiz da árvore de busca representa o problema original a ser resolvido. A partir desse momento, as regras do algoritmo são aplicadas para cada subproblema até que não existam mais nós a serem explorados. Um subproblema associado a um nó, que é escolhido por uma regra de seleção, ou é resolvido diretamente, ou é decomposto em dois ou mais nós, os quais são adicionados à árvore. Para cada nó, é computado o valor do limite superior (ou inferior), que identifica a melhor solução que aquele nó pode gerar. Caso esse valor seja pior que a melhor solução já encontrada, o nó é eliminado (*fathomed*), pois nem ele e nem seus descendentes podem produzir uma solução ótima para o problema.

As etapas do método branch and bound para determinar uma solução inteira ótima para um modelo de maximização (com <= restrições) podem ser resumidas da seguinte forma (http://web.tecnico.ulisboa.pt/mcasquilho/compute/_linpro/TaylorB_module_c.pdf).

1. Encontre a solução ótima para o modelo de programação linear com a restrição de números inteiros relaxada.
2. No nó 0 (raiz), seja a solução relaxada o limite superior e a solução inteira arredondada para baixo o limite inferior.
3. Selecione a variável com a maior parte fracionária para ramificação (*branching*). Crie duas novas restrições para essa variável, refletindo os valores inteiros particionados. O resultado será uma nova restrição $\leq$ e uma nova restrição $\geq$.
4. Crie dois novos nós, um para a restrição $\leq$ e outro para a restrição $\geq$.
5. Resolva o modelo de programação linear relaxado com a nova restrição adicionada em cada um desses nós.
6. A solução relaxada é o limite superior em cada nó, e a solução inteira máxima existente (em qualquer nó) é o limite inferior.
7. Se o processo produzir uma solução viável inteira com o maior valor limite superior dentre qualquer nó final, a solução inteira ótima foi atingida. Se uma solução inteira viável não aparecer, ramifique o nó com o maior limite superior.
8. Volte ao passo 3.

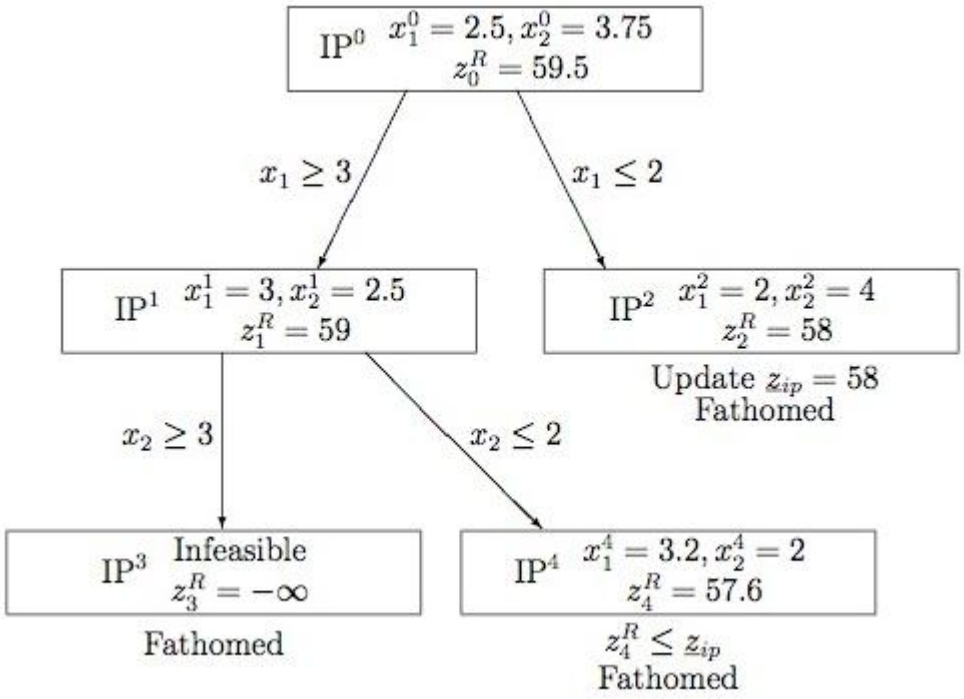
Para um modelo de minimização, soluções relaxadas são arredondadas, e limites superior e inferioros são invertidos.

Problemas de programação linear inteira mista também podem ser resolvidos usando o método B&. As mesmas etapas básicas aplicadas ao modelo de variáveis inteira são usadas para um modelo inteiro misto com apenas algumas diferenças.

Primeiro, no nó 0, apenas as variáveis restritas a números inteiros são arredondadas para baixo para atingir o limite inferior. Segundo, ao determinar de qual variável derivar, selecionamos a maior parte fracionária entre apenas aquelas variáveis que devem ser inteiras. Todos os outros passos permanecem os mesmos. A solução ótima é atingida quando uma solução viável é gerada em um nó que possui valores inteiros para as variáveis que requerem números inteiros e que atingiu o maior limite superior de todos os nós finais (nós folhas).

Exemplo 1 – PLI

$$\begin{aligned} \max Z = \quad & 13x1 + 8x2 \\ \text{sa} \quad & x1 + 2x2 \leq 10 \\ & 5x1 + 2x2 \leq 20 \\ & x1, x2 \geq 0 \end{aligned}$$



** *fathomed*: totalmente resolvido, totalmente examinado, explorado

Exemplo 2 – PLB

$$\text{maximize } Z = 8x1 + 11x2 + 6x3 + 4x4$$
$$\text{as} \quad 5x1 + 7x2 + 4x3 + 3x4 \leq 14$$
$$xi \in \{0, 1\}$$

Nó	Nível	Restr. Ad.	Solução - Tipo	Z	x1	x2	x3	x4	
0	0		Não Inteira	22,00	1	1	0,5	0	Branch
0.1	1	x3 ≤ 0	Não Inteira	21,66	1	1	0	0,67	Branch
0.2	1	x3 ≥ 1	Não Inteira	21,85	1	0,71	1	0	Branch
0.2.1	2	x2 ≤ 0	Sub-ótima	18,00	1	0	1	1	fathomed
0.2.2	2	x2 ≥ 1	Não Inteira	21,80	0,6	1	1	0	Branch
0.2.2.1	3	x1 ≤ 0	INTEIRA	21,00	0	1	1	1	fathomed
0.2.2.2	3	x1 ≥ 1	Inviável	NaN	1	1	1		fathomed
0.1.1	2	x4 ≤ 0	INTEIRA	19,00	1	1	0	0	fathomed
0.1.2	2	x4 ≥ 1	Não Inteira	21,42	1	0,86	0	1	Branch
0.1.2.1	3	x2 ≤ 0	Sub-ótima	12,00	1	0	0	1	fathomed
0.1.2.2	3	x2 ≥ 1	Não Inteira	21,40	0,8	1	0	1	Branch
0.1.2.2.1	4	x1 ≤ 0	Sub-ótima	15,00	0	1	0	1	fathomed
0.1.2.2.2	4	x1 ≥ 1	Inviável	NaN	1	1	0	1	fathomed

fathomed≡totalmente sondado, superado. Um nó totalmente sondado não gera ramos. A partir dele não serão gerados descendentes.

Dual Prices and Reduced Costs in Integer Programs

Dual prices and reduced costs in solution reports for integer programs have a restricted interpretation. For first time users of IP, it is best to simply disregard the reduced cost and dual price column in the solution report. For the more curious, the dual prices and reduced costs in a solution report are obtained from the linear program that remains after all integer variables have been fixed at their optimal values and removed from the model. Thus, for a pure integer program (i.e., all variables are required to be integer), you will generally find:

- *all dual prices are zero, and*
- *the reduced cost of a variable is simply its objective function coefficient (with sign reversed if the objective is MAX).*

For mixed integer programs, the dual prices may be of interest. For example, for a plant location problem where the location variables are required to be integer, but the quantity-shipped variables are continuous, the dual prices reported are those from the continuous problem where the locations of plants have been specified beforehand (at the optimal locations).

https://www.lindo.com/downloads/LINGO_text/Chapter11.pdf

http://www.inf.u-szeged.hu/~london/Linprog/linprog4.pdf

CUT-plane http://www.cs.tut.fi/kurssit/ELT-53656/lecture08_ELT53656.pdf

https://www.youtube.com/watch?v=Ndt2gp6CO6s&list=PLbxFfU5GKZz1Tm_9RR5M_uvdOXpJJ8LC3&index=23

https://www.youtube.com/watch?v=S8Uqojj60ak&list=PLbxFfU5GKZz1Tm_9RR5M_uvdOXpJJ8LC3&index=24

https://www.youtube.com/watch?v=GWYWewB5Qh4&list=PLbxFfU5GKZz1Tm_9RR5M_uvdOXpJJ8LC3&index=25